## CLAIMS

What is claimed is:

1. In a computer system, a method of defining sets of data to be retrieved from a data store, comprising the steps of:

providing a written representation of a desired data set in terms of dimensions and relation instances, the desired data set having a certain set type;

implying constraints on relation instances or dimensions based on the set type of the desired data set and dimension expressions, said step of implying constraints enabling length of the written representation to be minimized; and

using the written representation to query the data store to retrieve the desired data set.

2. A method as claimed in Claim 1 wherein the step of providing a written representation includes employing any combination of a disjunctive expression and a conjunctive expression; and

further comprising the step of:

using a record operator to apply multiple atomic constraints on a same tuple in a relation.

3. A method as claimed in Claim 1 wherein the step of providing a written representation includes employing any combination of a disjunctive expression and a conjunctive expression such that the written expression has one or more conjuncts; and

further comprising the steps of:

determining which relations to use in the evaluation of each conjunct in the written representation by using (i) the set type of the desired data set, (ii) a dimension list based on records in the conjunct of the written representation of the data set, and (iii) dimension relation associations and related inclusion criteria.

4. A method as claimed in Claim 1 wherein the step of providing a written representation includes employing any combination of a disjunctive expression and a conjunctive expression; and

further comprising the steps of:

performing OR-distribution on disjunctive expressions; and

eliminating from disjunctive expressions, conjuncts with undefined binding variables.

5. A method as claimed in Claim 1 wherein the step of providing a written representation includes employing any combination of disjunctive expressions and conjunctive expressions; and

further comprising the steps of:

translating conjunctive expressions to respective SQL joint terms; and

translating disjunctive expressions to respective SQL-union terms.

6. A method as claimed in Claim 5 wherein the step of translating a conjunctive expression includes reordering terms of a conjunct;

further comprising the steps of:

moving terms with negation and no binding variables furthest back in the conjunct;

placing terms without negation ahead of other terms in the conjunct, and

adding a terms with the certain set type of the desired data set to the front of the conjunct if all existing terms have negation.

7. A method as claimed in Claim 5 further comprising the step of rewriting the disjunctive and/or conjunctive expressions such that an SQL union operator is

applied after the SQL join terms are calculated, resulting in a computationally faster implementation.

8. A method as claimed in Claim 1 further comprising the step of automatically enforcing a record-operator where an expression in the written representation without the record-operator is semantically equivalent to the expression with the record-operator.

9. A method as claimed in Claim 8 further comprising the step of automatically enforcing a record-operator based on attribute structure information or a greedy record enforcement based on attribute coexistence in relations.

10. A method as claimed in Claim 1 wherein the step of providing a written representation includes employing an IN-statement and a disjunctive expression in a nested set; and

further comprising the step of applying OR-distribution within the nested set by treating the IN-statement effectively as a record-operator expression.

11. A method as claimed in Claim 1 wherein the data store has a native query engine; and

further comprising the step of rewriting the written representation such that upon translation of the rewritten written representation into code for the native query engine, the code is optimized for querying the data store.

12. A method as claimed in Claim 1 wherein the step of providing a written representation includes utilizing a certain symbol to specify hierarchical constraints on dimensions.

13. A method as claimed in Claim 12 wherein the certain symbol is a colon or an equal sign followed by a colon.

14. A method as claimed in Claim 1 wherein the step of providing a written representation includes utilizing certain symbols to specify constraints on dimensions that are mapped according to domain type of said dimension.

15. A method as claimed in Claim 1 wherein the step of providing a written representation includes providing an expression with aggregate function; and further comprising the step of:

translating the expression into a SQL statement in which a GROUP BY clause is implicitly defined by the certain set type.

16. A method as claimed in Claim 1 wherein the step of providing a written representation includes providing an expression with aggregate function; and further comprising the steps of:

translating calculated expression with aggregate functions in records into a SQL HAVING clause and expression without an aggregate function into a SQL WHERE clause.

17. A method as claimed in Claim 1 wherein the step of providing a written representation includes using an expression with bidirectional inlining of SQL statements and SDL statements; and

further comprising the steps of

denoting a start of SDL inlining in SQL with a bracket type symbol or

referring to SDL metadata in an SQL statement by enclosing it in a bracket-operator.

18. A method as claimed in Claim 17 wherein the step of denoting includes denoting the start of SQL inlining in SDL with a square-bracket followed by a certain keyword

or

denoting the start of SQL inlining in SDL with a curly-bracket followed by a certain keyword.

19. A method as claimed in Claim 1 wherein the step of providing a written representation includes providing an expression that is a combined statement having bidirectional inlining of SQL statements and SDL statements; and

further comprising the steps of:

using a pass-through compiler, compiling the combined statement including (i) ignoring pure SQL code, (ii) recursively compiling any inlined SDL statements into pure SQL by feeding any SDL statement containing SQL code recursively into the pass-through compiler, and (iii) feeding any pure SDL statement into an SDL-to-SQL compiler.

20. A method as claimed in Claim 1 wherein the step of providing a written representation includes using working expressions with sets of unspecified output type; and

further comprising the steps of:

for a given working expression, implying output type from type of dimension in a preceding expression.

21. A method as claimed in Claim 1 wherein the step of providing a written representation includes using an expression with incomplete names of dimensions;

further comprising the steps of:

resolving full dimension names by using a dimension prefix determined by the set type and current scope.

22. A method as claimed in Claim 1 wherein the step of providing a written representation includes using an expression with virtual dimensions;

further comprising the steps of

resolving virtual dimensions to a dimension name by constructing said name from defined dimension names, separators, and a part of other dimension names; and

generating a corresponding relation for a record expression to apply constraints of one or more virtual dimensions through a join of available relations, based on the dimension name and a corresponding defined dimension.

23. A method as claimed in Claim 1 wherein the step of providing a written representation includes utilizing a syntax and semantic aware editor that makes semantic checks based on domain types and indicates incorrect expressions.

24. A method as claimed in Claim 23 wherein the editor includes a description mode enabling a user to visualize SDL expressions for forming the written representation based on dimension description metadata.

25. A method as claimed in Claim 1 wherein the step of providing a written representation is facilitated by a user-interactive dialog, that includes:

generating an SDL expression which depends on known fields having input provided by a user at any given time; and

eliminating terms with undefined constraint dimensions.

26. A method as claimed in Claim 1 wherein the step of providing a written representation is facilitated by a data browser, the data browser supporting nested dimension drilling for generating virtual dimensions or nested set definitions.

27. A method as claimed in Claim 26 wherein the data browser shows only dimensions related to a selected dimension.

28. A method as claimed in Claim 1 further comprising the step of supporting federated queries by any combination of: a) using prefixes or aliases to map dimension names and b) having session dependent federation in which metadata is defined at runtime and federation nicknames are generated on demand.

29. A method as claimed in Claim 1 wherein the method allows URL templates based on dimension and domain fingerprints for reports.

30. A method as claimed in Claim 1 wherein the steps of providing and implying are implemented in SDL and uses one of an object-relational database, an object-oriented database or a memory based system to implement the SDL language.

31. In a computer system, apparatus for defining sets of data to be retrieved from a data store, comprising:

an input component for providing a written representation of a desired data set in terms of dimensions and relation instances, the desired data set having a certain set type; and

an assembly coupled to receive the written representation, in response the assembly implying constraints on relation instances or dimensions by one of the set type of the desired data set and dimension expressions, said implying constraints enabling length of the written representation to be minimized.

32. Apparatus as claimed in Claim 31 wherein the written representation includes any combination of a disjunctive expression and a conjunctive expression; and

the assembly further performs at least one of

(a) OR-distribution on disjunctive expressions and eliminates from disjunctive expressions, conjuncts with undefined binding variables,

(b) application of multiple atomic constraints on a same tuple in a relation, and

(c) determination of which relations to use in the evaluation of each conjunct in the written representation by using (i) the certain set type, (ii) a dimension list based on records in the conjunct and (iii) dimension relation associations.

33. Apparatus as claimed in Claim 31 wherein the written representation includes any combination of disjunctive expressions and conjunctive expressions; and

the assembly translates conjunctive expressions to respective SQL join terms and translates disjunctive expressions to respective SQL-union terms.

34. Apparatus as claimed in Claim 33 wherein the assembly rewrites the disjunctive and/or conjunctive expressions such that the SQL union operator is applied after the SQL join terms are calculated, resulting in a computationally faster implementation.

35. Apparatus as claimed in Claim 33 wherein the assembly further reorders terms of a conjunctive expression.

36. Apparatus as claimed in Claim 31 wherein the assembly automatically enforces a record-operator where an expression in the written representation without the record-operator is equivalent to the expression with the record-operator.

37. Apparatus as claimed in Claim 31 wherein the written representation includes an IN-statement and a disjunctive expression in a nested set; and

the assembly applies OR-distribution within the nested set by treating the IN-statement effectively as a record-operator expression.

38. Apparatus as claimed in Claim 31 wherein the data store has a native query engine; and

the assembly further translates the written representation into code for the native query engine in a manner such that the code is optimized for querying the data store.

39. Apparatus as claimed in Claim 31 wherein the written representation utilizes a certain symbol to specify hierarchical constraints on dimensions.

40. Apparatus as claimed in Claim 31 wherein the certain symbol is a colon or an equal sign followed by a colon.

41. Apparatus as claimed in Claim 31 wherein the written representation includes an expression with aggregate function; and

the assembly translates said expression into an SQL statement in which a GROUP BY clause is implicitly defined by the certain set type.

42. Apparatus as claimed in Claim 31 wherein the written representation includes an expression with aggregate function; and

the assembly translates calculated expressions with aggregate functions in records into an SQL HAVING clause, and translates expressions without an aggregate function into an SQL WHERE clause.

43. Apparatus as claimed in Claim 31 wherein the input component includes:

an editor for composing written representations; and

a data browser for enabling user browsing of dimension values and relations of the data store, to assist a user in composing desired written representations.

44. Apparatus as claimed in Claim 43 wherein the data browser provides at least one of graphical views of dimension hierarchies for user browsing, and nested dimension drilling for generating virtual dimensions or nested set definitions.

45. Apparatus as claimed in Claim 43 wherein the editor includes any combination of:

semantic checks based on domain types;

a description mode enabling a user to select SDL expressions for forming the written representation based on dimension description data; and

a dialog for generating an SDL expression as a function of user input into certain fields.

46. Apparatus as claimed in Claim 45 wherein the editor eliminates terms with undefined constraints on dimensions.

47. Apparatus as claimed in Claim 43 wherein the editor employs a user interface which supports drag and drop of dimensions and relation values in written representations being composed.

48. Apparatus as claimed in Claim 31 wherein the written representation includes an expression with bidirectional inlining of SQL statements and SDL statements.

49. Apparatus as claimed in Claim 48 wherein said expression denotes SDL portions in an SQL statement with one bracket operator, and denotes SQL portions in an SDL statement with another bracket-type operator.

50. Apparatus as claimed in Claim 48 further comprising a pass-through compiler for compiling said expression by (i) ignoring pure SQL code, (ii) recursively compiling any inlined SDL statements into pure SQL by feeding any SDL statement containing SQL code recursively into the pass-through compiler, and (iii) feeding any pure SDL statement into an SDL-to-SQL compiler.

51. Apparatus as claimed in Claim 31 wherein the written representation includes working expressions with sets of unspecified output type; and

the assembly, for a given working expression, implies output type from type of dimension in a preceding expression.

52. Apparatus as claimed in Claim 31 wherein the written representation includes an expression with incomplete names of dimensions; and

the assembly resolves full dimension names.

53. Apparatus as claimed in Claim 31 wherein the written representation includes an expression with virtual dimensions; and

the assembly resolves virtual dimensions to a dimension name.

54. Apparatus as claimed in Claim 31 wherein the assembly supports federated queries of the data store by using prefixes or aliases to map dimension names.

55. Apparatus as claimed in Claim 54 wherein the assembly has session dependent federation in which metadata is defined at runtime and federation nicknames are generated on demand.

56. Apparatus as claimed in Claim 31 wherein the assembly provides URL templates based on dimension and domain fingerprints for reports.